REPORT NO. 466

COO-1469-0188

GRASS: Remote Facilities Guide

by

R. Haskin
J. Nickolls
M. Michel

July 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

REPORT NO. 466

GRASS: Remote Facilities Guide

by

R. Haskin
J. Nickolls
M. Michel

July 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS 61801

* This report supported in part by the Atomic Energy Commission under grant US AEC AT(11-1)1469.

Page

1. INTRODUCTION

This report describes facilities that are available to users, and to user written programs running on the remote computer, of the Graphical Remote Access Support System. Familiarity with the basic system and with operation of a terminal is assumed. Consult [1, 2].

The first section describes features that can be called during execution of general user programs on the remote computer. These calls enable convenient interactive communication between the programs and the terminal. The local program REACT must be executing on the PDP-8 during this time.

A facility for performing library file maintenance is described next. This is a program similar to a user program in that it runs on-line in the remote computer under GRASS. It communicates with the user at a terminal via the REACT program, and allows various library operations including saving, deleting, and fetching of files. An important aspect of this program is its use of XFILE macros to access the remote computer library and its use of S8 macros to transmit data to the PDP-8. These two capabilities are also available to user written programs. However, since they are supported by GRASS on a "lower level" than the communication facilities described in the first section, their correct use is very critical. That is, these macros call internal system functions directly; hence, incorrect use, for example, could destroy data in the library or cause transmission difficulties with the PDP-8. Use of these facilities is described in [3]; no use is permitted without prior testing under supervision. A batch test monitor is available for this purpose.

The last section describes procedures for producing hardcopy documentation of pictures and mnemonics stored in the library. The requests for hardcopy are generated on-line, but the actual plotting is done at a later time by an off-line (batch stream) job.

2. INTERACTIVE COMMUNICATION

The PDP-8/360 Interactive Communications Package is comprised mainly of two programs; REACT, the generalized communications program segment which runs in the PDP-8, and COMMUNE, a group of re-entrant routines which run in the 360 and are callable from user programs running under the GRASS system. It is assumed that the reader is already reasonably familiar with the use of REACT and the general operation of the graphics terminals. What follows is a full description of COMMUNE and its use from both FORTRAN and assembly language programs.

COMMUNE provides the user with routines to perform basic input/output functions with the terminals, including capability to receive joystick hit coordinate information, to receive text lines typed by the user, and to send text messages to the user. Also, the user has the capability of constructing the elements of pictures (line blocks, text blocks, etc.) and sending them to the PDP-8, either for display or for addition to the user's data structure.

2.1 Basic Input/Output Facilities

Three routines comprise the basic text-line-joystick-hit I/O package. These routines are MASK, REPLY, and MESAGE. Two other routines, FINDNM and NUMCHR, aid in interpreting the input from the terminal. Another routine, ERASE, allows the user to erase the display screen and display a frame at any time.

MASK

MASK provides the user with a way to filter undesired input from the terminal, and to control whether or not user text lines are echoed to the display as they are typed. COMMUNE filters input from a terminal into ten types; 1-9 are joystick hits in screen segments 1-9 (c.f. Figure 1), and input type 10 is a text line. This filtering can be set up by a call to MASK. The format of a call from FORTRAN is:

CALL MASK(IMASK)

IMASK is an INTEGER*4 array with 11 elements. These elements have meaning as follows:

IMASK(1)--1 = do not echo a user-typed text line on the display
0 = echo a user-typed text line

IMASK(2) through IMASK(10)--for screen segments 1-9
1 = accept joystick hit
0 = ignore joystick hit

IMASK(11)--
1 = accept a user-typed text line
0 = ignore a user-typed text line

REPLY

Input to the user's program from the terminal is performed by calls to REPLY. When REPLY is called, the next input from the terminal (after filtering as per previous calls to MASK) is returned to the program, along with information as to the input type and length. The FORTRAN calling sequence is:

CALL REPLY(ITYPE,LEN,IDATA,NCHARS)

IDATA is an INTEGER*4 array with 20 elements. The type and data contained in the input are determined as follows:

ITYPE--1 = input was a joystick hit
0 = input was a user-typed text line

LEN-- = # of elements of IDATA containing valid data (i.e., if a 12 character text line were returned, LEN would equal 3; for a joystick hit, LEN would also be 3)

IDATA--

For a Joystick Hit

IDATA(1) = X coordinate of joystick hit in increments (from 0 to 1023)

IDATA(2) = Y coordinate of hit (0-800)

IDATA(3) = screen segment of hit (1-9)

For a Text Line

IDATA(1)-IDATA(LEN) contain the returned EBCDIC characters packed four per array element. If the total number of characters in the typed text line was not a multiple of four, IDATA(LEN) is padded with blanks to fill out the element.

NCHARS-- number of text characters sent by PDP-8, before being padded with blanks by REACT. This parameter is optional and may be omitted if not needed.

MESSAGE

MESSAGE allows the user to send a text line to the display. The FORTRAN calling sequence is:

```
CALL  MESAGE(ITEXT,LEN)
```

ITEXT is an INTEGER*4 array having at least (LEN/4) + 1 elements and containing the EBCDIC characters to be displayed at the terminal, or, optionally, is a character constant. LEN is the number of characters in the message to be displayed. Sample calls are:

```
C
C   SAMPLE      #1
C
            DIMENSION ITEXT1(3)
            DATA ITEXT1/'SOME JUNK'/,LEN/9/
            CALL MESAGE(ITEXT1,LEN)
            ITEXT1(2)='MOR'
            ITEXT1(3)='E  '
            CALL MESAGE(ITEXT1,LEN)
            STOP
            END
C
C   SAMPLE      #2
C
            CALL MESAGE('SOME JUNK',9)
            CALL MESAGE('SOME MORE',9)
            STOP
            END
```

Both examples result in the lines

```
SOME JUNK
SOME MORE
```

being displayed on the terminal.

FINDNM, NUMCHR

Two routines exist which allow the user to convert numeric data in his text line to binary. The first of these, FINDNM, scans the array filled by REPLY and determines the location of numeric character strings in the array. NUMCHR is a function subroutine whose input is this character string. It converts the string into a fixed point number and returns it. The calling sequences are:

```
CALL  FINDNM(IDATA,IFIRST,ILAST)
```

IDATA is an INTEGER*4 array containing the EBCDIC character string in which the number is to be found (i.e., the IDATA array after a REPLY call). IFIRST and ILAST are the character positions in IDATA (from 1 to 72) between which the number is to be found. Upon return, IFIRST and ILAST contain the character positions of the first and last digits of the first numeric string encountered scanning left to right. If no numeric string is found, IFIRST=ILAST=Ø upon return.

ERASE

ERASE causes the display screen to be erased. Optionally, a frame can be put up also. The FORTRAN calling sequence is:

```
CALL  ERASE(IFRAME)
```

IFRAME = 0 for no frame
= 1 for frame to be displayed after erase

EXAMPLE

The code shown below will display a message on the user's terminal, to which he responds by typing in a number. The number will be converted to an integer value and stored in INUM.

```
          DIMENSION IMASK(11),ITEXT(20)
C
C   SET IMASK TO ECHO TEXT LINES & IGNORE J.S. HITS
C
          DATA IMAST/1Ø*Ø;1/
          CALL MASK(IMASK)
C
C   ASK USER TO TYPE IN A NUMBER
C
    69    CALL MESAGE('TYPE IN A NUMBER',16)
C
C   DECODE IT, ITYPE HAS TO BE Ø (TEXT)
C
    86    CALL REPLY(ITYPE,LEN,ITEXT,NCHARS)
          IFIRST = 1
          ILAST = NCHARS
          CALL FINDNM(ITEXT,IFIRST,ILAST)
C
C   CHECK IF WE HAVE A REASONABLE NUMBER
C
          IF(IFIRST.EQ.Ø.OR.ILAST.GT.IFIRST+8) GO TO 69

    INUM=NUMCHR(ITEXT,IFIRST,ILAST)
    .
    .
    .
    .
```

2.2 Data Structure Handling Facilities

COMMUNE allows the user to build and send to the PDP-8 line and text blocks. The calls to the routines involved look much like CalComp plotter calls, with the extension that they allow specification of coordinates in either integer or floating point. Options are available to allow the user to erase the display screen, put up a frame, to display or not display the data, and to replace the proper block in the user's current local (PDP-8) data structure.

LINE

LINE allows the user to add a line to his current line block. If the block was previously empty, a new one is initialized. The "beam" or "pen" position is moved to the specified coordinates, drawing a line if specified.

The X and Y coordinates may be specified either as real or integer values. If real values are supplied, the coordinates are assumed to represent inches on the screen. If integer values are supplied, the coordinates represent screen increments (about 124.1 increments per inch, 1024 increments across the width of the screen). The FORTRAN calling sequence is:

```
CALL  LINE(X,Y,INTEN)
```

or

```
CALL  LINE(X,Y,INTEN,&STMT)
```

Where:

X = X coordinate of line end (either integer or real)

Y = Y coordinate of line end

INTEN = 1--draw line from current beam position to (X,Y)
0--move beam without drawing to (X,Y)

&STMT = optional parameter, indicates the statement number to which control is to be passed (via a RETURN 1 statement) if the line block filled up as a result of this call.

LINED

LINED is similar to LINE, except X & Y represent differential values to be added to the current beam position to determine the new one. After a call, the new x position will be the previous x + X from this call.

TEXT

TEXT allows the user to add characters to the text block. If the block was previously empty, a new one is initialized. The FORTRAN calling sequence is:

```
CALL  TEXT(X,Y,ITEXT,LEN)
```

or

```
CALL  TEXT(X,Y,ITEXT,LEN,&STMT)
```

X,Y--fixed or floating coordinates of the text, as in LINE or LINED

ITEXT--either a literal constant or the name of an INTEGER*4 array containing the text packed 4 characters per element.

LEN--the number of characters in ITEXT to be added to the text block.

&STMT--optional, same as in LINE & LINED

If X = Y = 0 and LEN is positive, the specified text appears in a new line immediately below the previous one. If X = Y = 0 and LEN is negative, ABS(LEN) is the number of characters and the characters are added to the current line. If LEN = 0, a blank line is generated.

SEND

Completed blocks may be output to the user by a call to SEND. Various options are available regarding what is to be done with the blocks by the PDP-8, and whether or not the blocks are to be reinitialized. The FORTRAN calling sequence is:

```
CALL  SEND(IBLK,ICODE)

IBLK--denotes the block(s) to be sent
      IBLK=1 means send the line block
      IBLK=2 means send the text block
      IBLK=-1 means send all initialized blocks
      IBLK=0 has a special meaning which will be
           discussed later

ICODE--an INTEGER*4 array with 5 elements.  Selects
       the options to be used for the call.  The
       meanings are as follows:

ICODE(1)--1 = reinitialize (make empty) all
              specified blocks when done sending
          0 = retain blocks after sending

ICODE(2)--1 = replace block(s) sent in the user's
              PDP-8 data structure
          0 = retain current PDP-8 data structure

ICODE(3)--1 = do not display the block(s)
          0 = display

ICODE(4)--1 = erase screen before displaying
              block(s)
          0 = add block(s) sent to current
              screen image

ICODE(5)--1 = put up frame before displaying block(s)
          0 = do not put up frame
```

Non-initialized (empty,) blocks are not sent.

An option is available which will cause the line block to be sent to the display and re-initialized if it fills up before another SEND call. To use this option, the block must not be marked to replace the current PDP-8 data structure, and must be for display. To do this:

```
ICODE(2) = 0
ICODE(3) = 0
SEND(0,ICODE)
   .
   .
   calls to LINE and LINED, possibly
   filling up several line blocks
   .
   .
SEND(1,ICODE)
```

This option behaves as follows:

(a) it remains in effect until the line block is sent explicitly (as by the second SEND call above),

(b) if more than one block is sent, only the first will cause an erase or frame,

(c) when the line block is finally sent explicitly, the ICODE parameters from the SEND(0,ICODE) are used (erase and frame behave according to (b),

(d) sending the line block explicitly clears the option.

Writing an interactive program to draw a circle at a specified location on the screen is now relatively trivial, and the code might look as follows:

```
      DIMENSION ICODE(5),IMASK(11)
C
C   SET IMASK TO ECHO TEXT LINES & ACCEPT J.S. HITS
C   IN DRAW AREA
C
      DATA IMASK/4*Ø,1,5*Ø,1/,FAC/.01744/
C
C   SET ICODE TO REINITIALIZE BLOCKS AFTER EACH SEND,
C   DISPLAY ONLY WITH ERASE AND FRAME
C
      DATA ICODE/1,Ø,Ø,1,1/
C
      CALL MASK(IMASK)
1     CALL MESAGE('TYPE IN RADIUS IN INCHES',24)
      CALL NUMBER(RADIUS,&1)
      IF (RADIUS.GT.3.) GO TO 1
2     CALL MESAGE('JOYSTICK CENTER',15)
      CALL HIT(X,Y,&2)
      INTEN=0
      DO 69 I=1,361
      SC=X+RADIUS*COS(I*FAC)
      YC=Y+RADIUS*SIN(I*FAC)
      CALL LINE(XC,YC,INTEN)
69    INTEN=1
      CALL SEND(1,ICODE)
      GO TO 1
      END
C
C   RETURN A NUMBER TYPED FROM THE KEYBOARD
C
      SUBROUTINE NUMBER(XNUM,*)
      DIMENSION ITEXT(20)
      COMMON ITEXT
      CALL REPLY(ITYPE,LEN,NCHARS)
      IF (ITYPE.NE.0) RETURN 1
      IFIRST=1
```

```
      ILAST=NCHAR
      CALL FINDNM(ITEXT,IFIRST,ILAST)
      IF (IFIRST.EQ.0) RETURN 1
      XNUM=NUMCHR(ITEXT,IFIRST,ILAST)
      RETURN
      END
C
C     RETURN COORDINATES OF A J.S. HIT
C
      SUBROUTINE HIT(X,Y,*)
      DIMENSION ITEXT(20)
      COMMON ITEXT
      CALL REPLY(ITYPE,LEN,ITEXT)
      IF (ITYPE.NE.1) RETURN 1
      X=ITEXT(1)/124.1
      Y=ITEXT(2)/124.1
      RETURN
      END
```

2.3 Extended Input/Output Facilities

The standard READ/WRITE statements of FORTRAN IV can also be used. Simply replace the unit designations 5 and 6 with 1 and 2. Then each line produced by a WRITE will be sent to the terminal and each line to be READ will originate as a typed line from the terminal. After every 32 lines sent by WRITE statements, the display screen will be erased and display of subsequent lines will begin at the top. Note, as an example, that dumping 2000 lines of calculations on the screen is obviously not an effective use of this feature.

2.4 Creating a Module Using COMMUNE

Programs to be run from the graphics terminals using COMMUNE must be link-edited into the graphics load module library. They must also have CALLER, the routine to interface the user's FORTRAN program to COMMUNE, link-edited into the program and specified as the entry point. To do this, the user's deck should be set up as follows:

```
/*ID....
//  EXEC FORT
(FORTRAN source deck)
/*
//  EXEC LKED
  ENTRY CALLER
  NAME <name>(R)
/*
```

<name> is the name by which the program is to be referred when it is started from the terminal with a !S command (i.e., !S <name>). To allow use of the FORTRAN READ/WRITE facility, insert the card

```
INCLUDE SYSLIB(X#FIØCS)
```

before the ENTRY card.

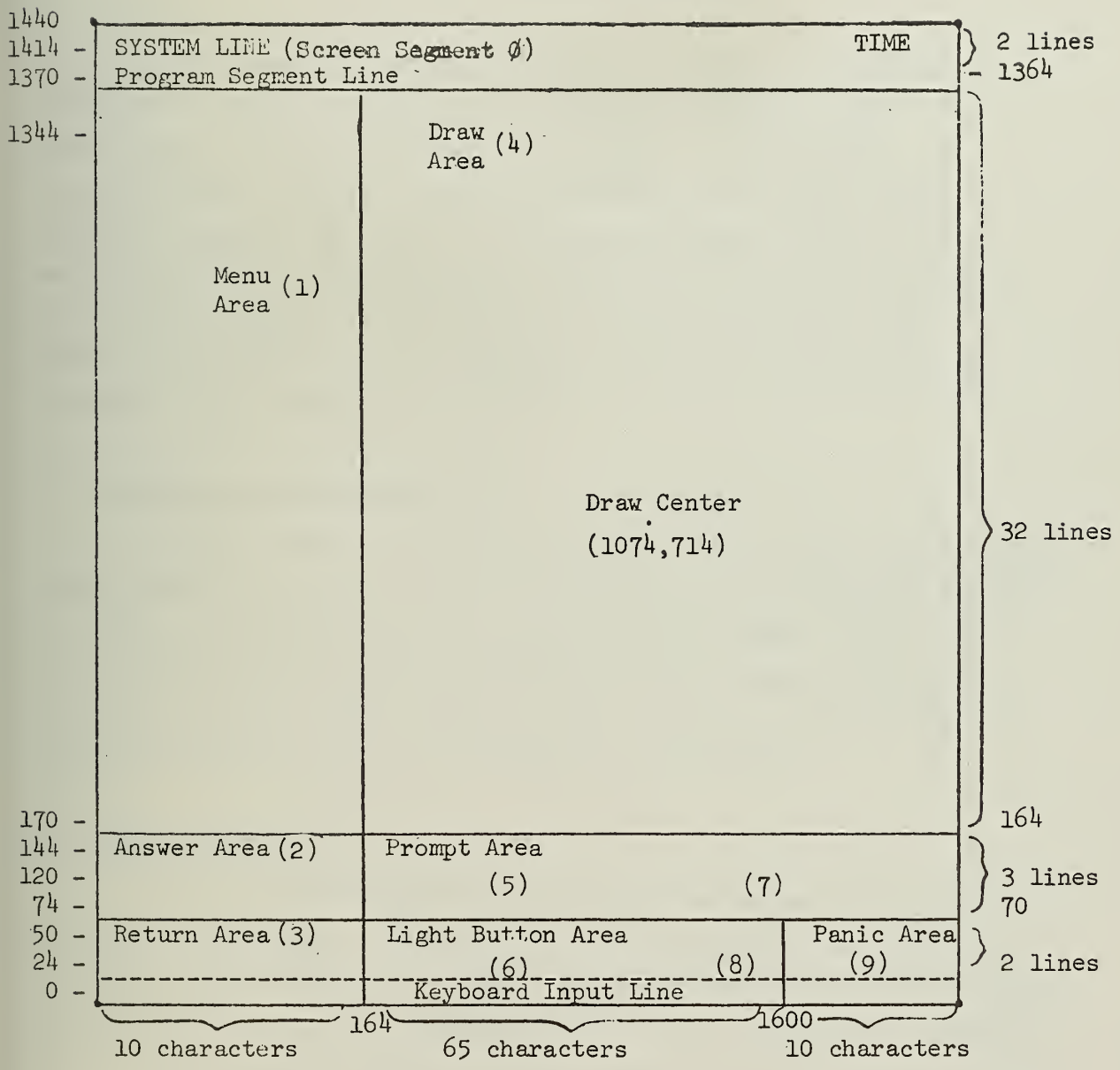


Figure 1. Display Screen Sectioning Detail

3. LIBRARY MAINTENANCE PROGRAM

The graphics library maintenance program, LSD, has been developed to provide facilities for storage, retrieval, and manipulation of data structures in the graphics filing system. It allows pictures to be sent from the PDP-8 and saved in the remote computer library or fetched from the remote computer library, and sent to the PDP-8. Pictures can be copied within the remote library; there are provisions for shared access to files among users, and there is also a provision for allowing pictures to be plotted by the hardcopy output package, PØT (c.f. 3.4).

3.1 Invoking and Terminating LSD

LSD is started and operated from the PDP-8 remote communications program REACT (c.f. [2]). After the user logs on to the remote computer via REACT, he invokes execution of LSD with the command !S LSD. All lines subsequently typed are interpreted as LSD commands until LSD is terminated with the !HALT command. At that time, all open files are closed and necessary clean up procedures are performed.

3.2 Command Syntax

All of LSD's commands are uniquely identified by the first two letters of the command. Therefore, it is only necessary to type these two letters when typing the command. The catalogue function, for example, can be invoked by typing "CATALOG", "CA", or, for that matter, "CAGRXMMP".

Most commands require specification of a file name or a library name. DISPLAY, for example, needs to know what picture to display, and CATALOG needs to know which library catalog to display. The syntax for generalized library names and generalized file names is:

1. Picture name--(<picturename>)*

   A picture name is an alphanumeric string 1 to 6 characters long.

2. Mnemonic name--(<mnemonicname>)

   A mnemonic name is an alphanumeric string 1 to 6 characters long, followed by a dot, followed by a mnemonic identification number (0-7). Example:

   RESIST.2

3. User name--(<username>)

   An alphanumeric string 1 to 8 characters long.

4. Library name--(<libraryname>)

   An alphanumeric string 1 to 8 characters long.

5. Generalized library name--(<glibname>)

   [[<username>.]<libraryname>]

Examples

a. CATALOG
b. CATALOG PICLIB
c. CATALOG HASKIN.PICLIB

The defaults for specifying a library name are as follows:

i. if the username is omitted, the name used is the one by which the user has logged on (i.e. Ex. b)

ii. if the entire library name is omitted, the username is chosen as above, and the default library name used is PICLIB (i.e. Ex. a).

---

* In the syntax descriptions, brackets ([]) around an item indicate that it is optional and may be omitted, and braces ({}) indicate that a choice must be made among the items enclosed.

6. Generalized file name--(<gfilename>)

{<picturename> / <mnemonicname>}[/<glibname>]

Examples

d. PLOT RESIST
e. PLOT RESIST.Ø
f. PLOT RESIST.Ø/PICLIB
g. PLOT RESIST/HASKIN.PICLIB
h. COPY RESIST.2/PICLIB JUNK.3/JOHNDOE.JUNKLIB

As a rule, blanks are used as separators between command fields and, therefore, may not be imbedded in command, file, or library names. Also, when blanks are used to separate command fields, one and only one blank must be used (i.e. PLOT␣␣RESIST is not valid).

3.3 Commands

Saving of Items

SA[VE] <gfilename>

SAVE allows a picture or mnemonic to be sent from the PDP-8 and saved in the remote library. The procedure for doing this is to first type in the SAVE command, and then, after LSD replies "SAVE READY", type the REACT command "##<picturename>" to cause the picture to be send to the remote computer. The new picture or mnemonic replaces any previous occurrence in the specified library.

Deletion of Items

DE[LETE] <gfilename>

Deletes the picture or mnemonic instance in the specified remote library.

Retrieval of Items

FE[TCH] <gfilename>

Retrieves the specified item and sends it to the PDP-8 where it replaces the current picture in the data structure. LSD then replies with a message to indicate whether or not the item was found.

Display of Items

DI[SPLAY] <gfilename>

The action of DISPLAY is similar to that of FETCH, except that the item is only displayed instead of replacing the current data structure.

Copying Items

CO[PY] <gfilename1> <gfilename2>

The item specified by <gfilename1> is copied into <gfilename2>, anything previously in the second file being overwritten. COPY can be used to transfer files from one user's library to another's library.

Obtaining a Catalog

CA[TALOG] <glibname>

A catalogue of the items in the specified library are displayed in the menu area on the terminal. Should this list exceed 25 items, the rest may be obtained by typing "MORE". Any other response will exit from catalog mode, allowing another command to be typed. When all names have been displayed, "DONE" will be displayed and another command can be typed.

Automatic Local Save Option

AU[TO]
DA[UTO]

These commands turn on and off, respectively, the "automatic save" feature in the PDP-8. When AUTO is in effect and a picture is sent to the PDP-8 via a FETCH command, the picture is saved on the PDP-8's local disk library. DAUTO, which is in effect until an AUTO command is typed, prevents this from occurring.

Obtaining Hardcopy Output

PL[OT] <gfilename> [<option list>]

To obtain hardcopy for a picture, use the PLOT command. One or more options may be specified, and the options list may be continued on several lines by terminating each line with a comma. Options and defaults are explained in detail in section 4, HARDCOPY PRODUCTION. PLOT puts the name of the item, the options, and the time and date when the PLOT command was typed into a file in the remote library. When the hardcopy output program is run later as a batch job, the item will be plotted.

4. HARDCOPY PRODUCTION

An offline package is available for producing hardcopy CalComp plots of pictures created with the GRASS system. The Plotting Offline for Terminals, PØT, facility is activated in two steps. Only plots of items stored in the remote computer section of the GRASS filing system library can be made, so the terminal user must first invoke the REACT program in the local computer and start LSD in the remote computer. Any pictures or mnemonics not already sent to the remote computer must be transmitted by the SAVE command. Then the user can issue the LSD PLOT command for each item to be plotted (c.f. section 3 concerning LSD and its PLØT command)

Second, a batch stream PØT job is run on the remote computer periodically during the day. This job retrieves all plot requests made by terminal users up to that time and plots the desired items on the CalComp plotter. PLØT commands can also be given to PØT on data cards during batch execution. These are plotted first, then the stored requests from terminal users are satisfied.

4.1 Step One: The PLOT Command

The PLOT command of LSD has the following format:

PLOT_FILENAME_OPT1,OPT2,...,OPTN

The filename can appear in a variety of formats (c.f. LSD, section 3). The filename may be a picturename or a mnemonicname (including the dot and mnemonic number). The file named must be in the graphics filing system, and all mnemonics referenced by a picture must be filed also. All pictures and mnemonics must be filed in the 360 under the same name that they have locally in the PDP-8.

The options specify what parts of the data structures in the file are to be plotted. Certain basic options are defaulted to ON; therefore, the user need not specify any options if he so desires.

The options currently available are listed below. Each option word is five characters long and begins with a one letter prefix describing what it affects. Any option may be preceded by a negation symbol "¬" to indicate the negative sense of the option. Starred options are defaulted to ON.

P-options concern the entire picture:

*PLLIN - Draw local lines from line block
*PCTXT - Print comment text from text block
*PSUBP - Draw subpictures from mnemonic instance block
*PTERM - Draw terminals and connections from terminal block
 PDECL - Print declarations
 PEQPM - Print equations and parameters
*PFRAM - Draw frame as seen on console
*PFTXT - Print system data in the frame

M-options concern mnemonic storage blocks:

*MLLIN - Draw local lines of mnemonics
 MCTXT - Print comment text of mnemonics
 MTERM - Draw terminals (when only drawing the mnemonic)
*MPARM - Print model parameter list beside mnemonic
 MTNM# - Print type name number on mnemonic terminals
 MTNML - Print mnemonic type name list in menu area

T-options concern terminal blocks:

 TSEQ# - Print sequence number on terminals
 TTNM# - Print type name number on terminals
 TTNML - Print type name list in menu area
 TDINV - Draw "invisible" connections between terminals

S-options concern subpicture instance blocks:

 SPARM - Print parameter assignments of subpicture instances

The PLOT command invokes a filename scanner and the plot option scanner PLTSCN. Thus, any syntax errors are caught when the command is typed in. If there are no errors, the message "PLOTTED" will appear on the screen, otherwise, an error message will appear containing the text at fault. The user may specify as many options on as many lines as needed by ending each constituent line with a comma. The last line must not have a comma at the end, or the next line typed will be interpreted as a continuation of the PLOT command. Blanks between parameters are ignored. Some examples follow:

1. PLOT TRNSIS PDECL,PEQPM,←MPARM,TSEQ#
2. PLOT RESIST.O/POOHBEAR.PICLIB MCTXT,MTERM, [firstline]
   MTNM#,MTNML . [secondline]
3. PLOT HEART/NICKOLLS.BIOLIB
4. PLOT NETWRK/HASKIN.CSLIB SPARM,←PCTXT

4.2 Step Two: POT Package

During the day, the program POT is run from batch. At this time, all the requests that were entered into the system via PLOT commands are actually acted upon.

If the PDP-8 section of the system is inoperative, or if additional requests are needed, PLØT commands can be entered via data card input to POT. The data cards look exactly like a PLOT command without the word "PLOT". Leading blanks are ignored, as are blanks between parameters. Multiple cards are allowed as on the console; just end each card except the last with a comma. Filenames on data cards must be fully qualified names (username and libraryname specified).

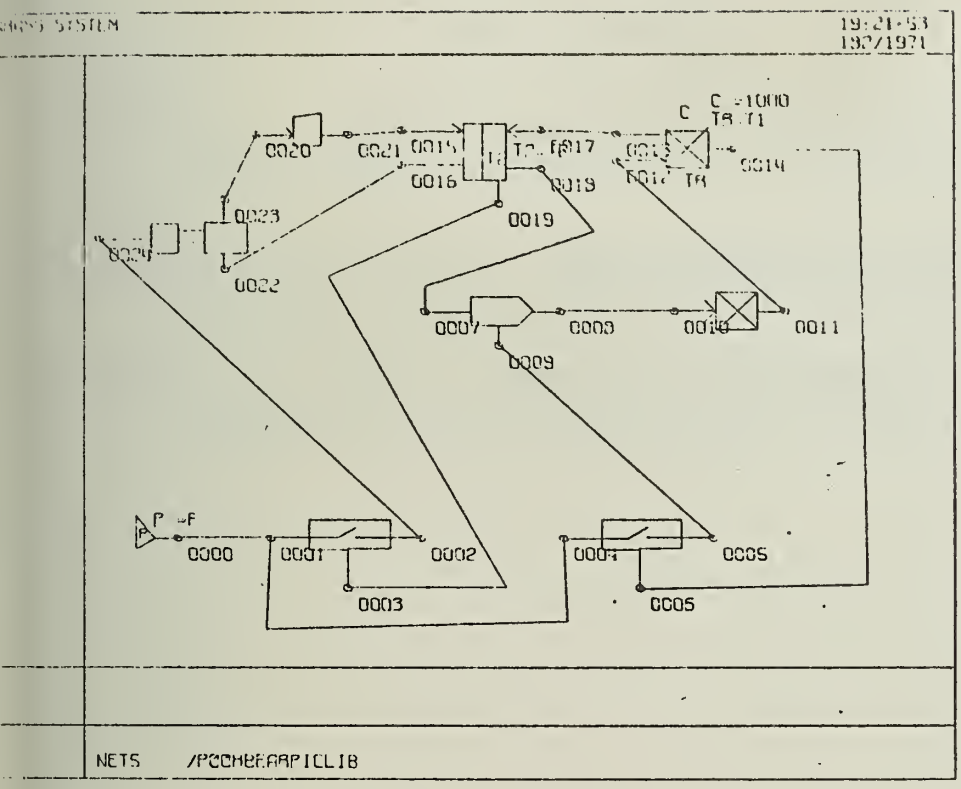
Each plot request results in one page of printed output in addition to the plot. This contains the name of the file printed, error messages, and other pertinent information.

Certain errors are nonfatal, such as a missing mnemonic in the filing system. A message is printed, and the rest of the plot is

completed. Other errors, such as missing pictures or illegal syntax on data cards cause a plot request to be flushed with an appropriate message. Filing system or I/O errors cause termination of the entire POT job.

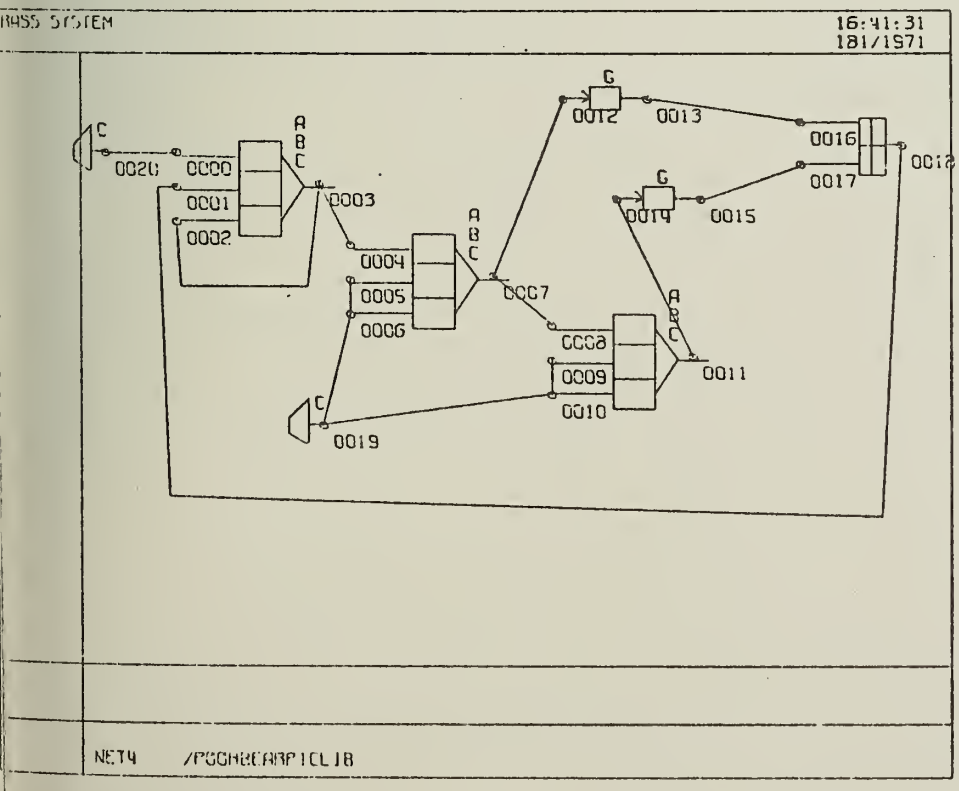
After completion of all plot requests, the file containing them is destroyed. This cannot interfere with terminal users doing PLOT commands while POT is executing. LSD and POT both use a 360 system ENQ macro, thus, insuring serial usage of the file. Hence, no plot requests are lost.

The following figure is an example of plots produced by the package.



EQUATIONS:
F=MIN(V5, T1*A

PARAMETERS:
T1
T2
V5



DEFINITIONS:
D

EQUATIONS:
D=SIN(TIME/10)

SAMPLE HARDCOPY PRINTS OF TWO MODELING SYSTEM NETWORKS

LIST OF REFERENCES

[1] Michel, M. J. GRASS: System Overview, Department of Computer Science Report No. 465, University of Illinois, Urbana, Illinois 61801, July 1971.

[2] Michel, M. J. and Koch, J. GRASS: Terminal User's Guide, Department of Computer Science Report No. 467, University of Illinois, Urbana, Illinois 61801, August 1971.

[3] Michel, M. J. GRASS: System Software Description, Department of Computer Science Report No. 468, University of Illinois, Urbana, Illinois 61801, August 1971.

AEC–427
(6/68)
CM 3201

U. S. ATOMIC ENERGY COMMISSION

UNIVERSITY–TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

AEC REPORT NO.	2. TITLE
COO-1469-0188	GRASS: Remote Facilities Guide

TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

REASON FOR RECOMMENDED RESTRICTIONS:

SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear, Professor
and Principal Investigator

Organization

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

Signature	Date
Charles W Gear.	August 1971

FOR AEC USE ONLY

AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.